Object Oriented Programing
Project Report

Credit: Photo by Candace McDaniel on StockSnap

Restaurant Management System

Prepared by

Joseph Oladunmoye
at

Seneca college

August 2024

Table of Contents

Assignment Details

Rubric:

Task/Criteria	Weight	Description
Task 1: Design & Coding (restaurant management system)	**30%**	Coding, quality of code, validations, OO principles, Error handling, completeness, executable without error, class diagrams, code documentation
Task 2: Conceptual & Report (Performance Analysis)	**30%**	Analysis and recommendations, trade-offs discussion, completeness, executable without error, documentation, references
Task 3: Conceptual & Research (Security Evaluation)	**30%**	Evaluation and analysis, completeness, executable without error, documentation, demo & presentation
General Deliverables	**10%**	Coding files (.cpp/.h), word document containing explanation on code, uml class diagram, output screenshots, conceptual questions. Text file with presentation link.

Deliverables:

1) Coding Files (.cpp/.h only)
2) Word document -- containing explanation on code + UML class diagram + output screenshots + answer to conceptual questions
3) A text file containing presentation link -- youtube

Task 1 (Design & Coding): Imagine you are tasked with designing an interactive menu for a new restaurant that utilizes object-oriented software development principles. Your goal is to create a user-friendly interface that allows customers to view the menu, place orders (include invoice, bill, tip, tax, etc), and customize their selections. Consider how you can utilize concepts such as inheritance, polymorphism, and encapsulation to optimize the design of the menu system. Your submission should include a detailed diagram illustrating the class hierarchy and relationships, along with a written explanation of your design decisions.

Task 2 (Conceptual & Report): You have been hired as a software consultant to evaluate the performance of a restaurant system designed in Task1. Your task is to analyze the system's efficiency, scalability, and reliability. Consider the data structures and algorithms employed in the system and assess their impact on performance. Develop a detailed report outlining your findings, recommendations for improvement, and any potential trade-offs involved. Support your analysis with real-life examples and provide a reference list of relevant resources that you consulted during your research.

Task 3 (Conceptual & Research): In this assignment, you will evaluate the security measures implemented in a restaurant management system to ensure the safety of customer transactions. Analyze the system's authentication mechanisms and potential vulnerabilities. You are encouraged to conduct a mock scenario where you attempt to identify and exploit security weaknesses in the system. Create a video presentation summarizing your evaluation, including a demonstration of any successful security breaches and a discussion of countermeasures to prevent future attacks.

I Task 1 (Design and Coding)

The project is an assignment project for OOP345 and will be implemented using Object Oriented Programming in C++. It will focus on inheritance, polymorphism, encapsulation and error handling to optimize the design of the menu system.

1 Restaurant Use Case

The project aims to design an interactive menu for a new restaurant that utilizes object-oriented software development principles. It will create a user-friendly interface that allows customers to view the menu, place orders, and customize their selections. There are 2 actors in the system, customer and administrator.

The customer interacts with the system to view menus, place orders, and manage billing. The customer can view the main menu to explore the available appetizers, entrees, desserts, drinks, and combos. The customer can also place an order by selecting items from the menu, adding a tip, and paying for the items. After order selection, the customer can view the bill for their order, review the total amount and proceed to checkout. The administrator is responsible for managing orders and editing the drinks and meals lists in the system.

2 Class Design Overview

The design implements a hierarchical structure consisting of several classes to represent different restaurant items and customer/admin operations. All classes use encapsulation to protect data attributes and provide public getters and setters to access data.

The base class is “RestuarantItem” composed of attributes and methods common to all drinks and meals. It also has a pure virtual function to be customized by child classes.

The “DrinkItem” and “MealItem” classes inherit from “RestaurantItem” and have additional data members and methods for customization e.g., “setPortionSize()” for the “MealItem” class.

Next in the hierarchy are “DessertItem”, “EntreeItem” and “SideDishItem” which inherit from “MealItem”. These have additional data members and methods specific to the type of meal e.g., “isVegetarian()” for the “EntreeItem” class.

A separate class “MenuManager” is used to manage the restaurant items. It comprises several container data members with methods to add or remove items. Polymorphism is demonstrated by using pointers to the “RestaurantItem” to refer to objects of its derived classes, thus allowing for dynamic management of objects.

3 Class Descriptions

3a RestaurantItem (Base Class)

The base class “RestaurantItem” is the parent class for all meal and drink categories. This class has attributes common to all derived classes such as name, id, description and price. Default constructors, getters and setters are defined in the “RestaurantItem” class to be inherited by the derived classes. The RestaurantItem has 3 child classes:

- RestuarantItem => DrinkItem
- RestuarantItem => MealItem
- RestuarantItem => LinkedList

The RestaurantItem class is designed as an Abstract class with displayDetails() implemented as a pure virtual function to be customized by the derived classes.

3b DrinkItem Class (Derived Class)

The drinkItem has data members that are specific to drinks such as size, alcohol content etc. Constructors, destructors, setters and getters are also defined. The displayDetails() function inherited from the restaurantItem class is redefined.

3c MealItem Class (Derived Class)

The mealItem contains data members that are specific to meals such as ingredients, portion size and allergens. Constructors and Destructors are defined in the mealItem class along with setters and getters. There are 3 meal categories derived from MealItem:

- MealItem => EntreeItem
- MealItem => DessertItem
- MealItem => SideDishItem

3d EntreeItem, DessertItem and SideDishItem (Derived Classes)

Each of these classes have data members that are specific to their category. For instance, the DessertItem has a data member “containsNuts”, a Boolean to identify desserts with nuts. The EntréeItem class has Boolean members such as “isVegetarian”, “isSpicy”, “isHandheld”. These data members are integrated into constructors as required.

3e MenuManager (Base Class)

This class is utilized to manage the collection of RestaurantItems. It has container data members consisting of vectors, maps and linked lists to hold various listing of meal and drink items including menus and inventory lists.

There are two static LinkedList data members which hold all inventory for meals and drinks. This is made static to ensure that there is a single instance of both lists shared across the application, centralizing the management of the drinksList and mealsList maintaining consistency in state.

Four menus are defined as vector objects: EntreeMenu, “drinksMenu”, “dessertMenu” and “sideDishMenu”. These vectors have a protected status so that they can be directly accessed by another derived class “Customer” which manages customer orders.

Getters and setters in the MenuManager class are used to provide functionality to add to various types of container items for different uses. For example, there is a add function for the linkedlist to add meal items and another add function for different menu options for vectors.

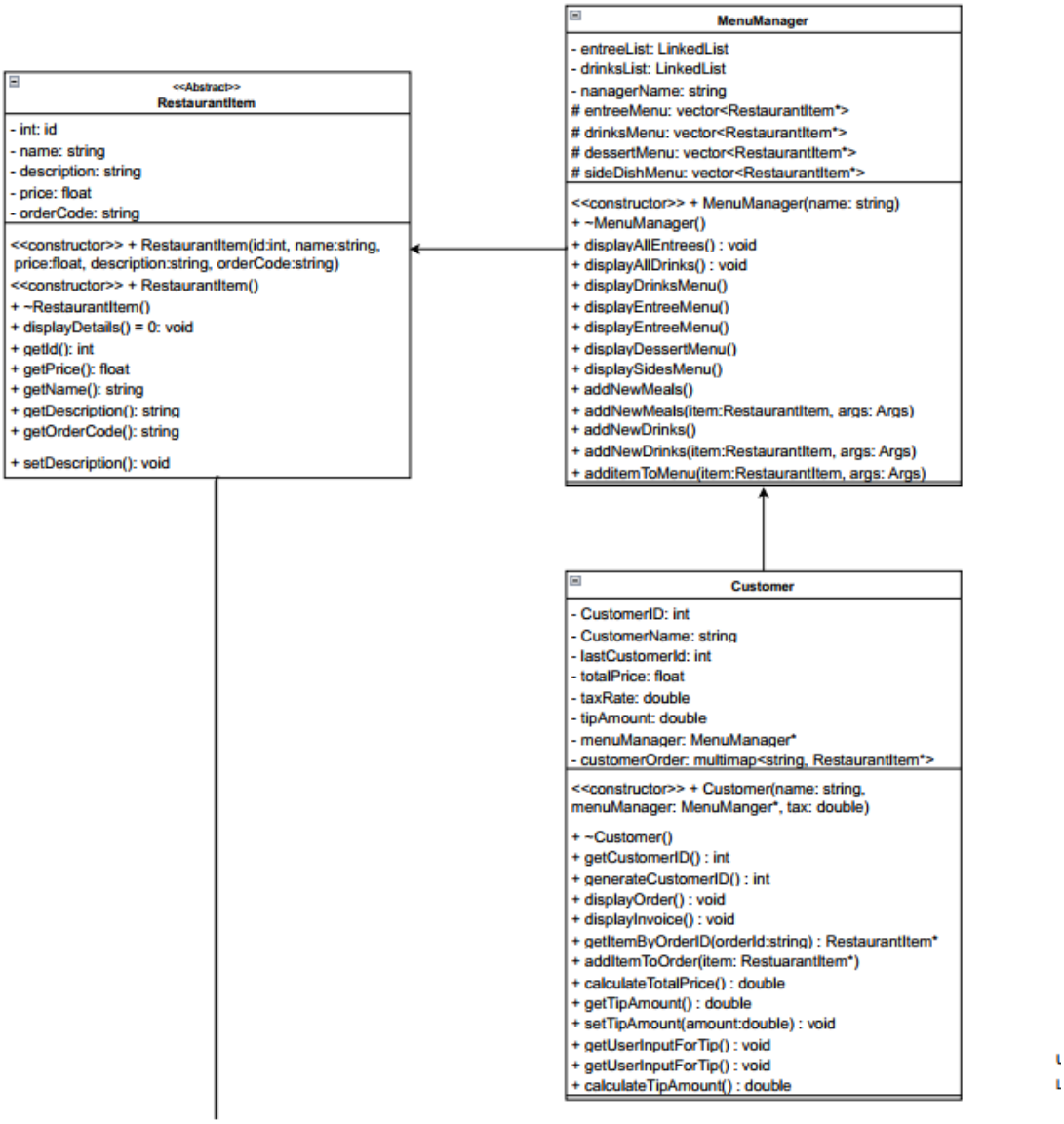
Several search functions are defined which work between different data structures. For instance, items are searched from the Linkedlist and added to vector objects. AS used in populating various menus from the overall lists.

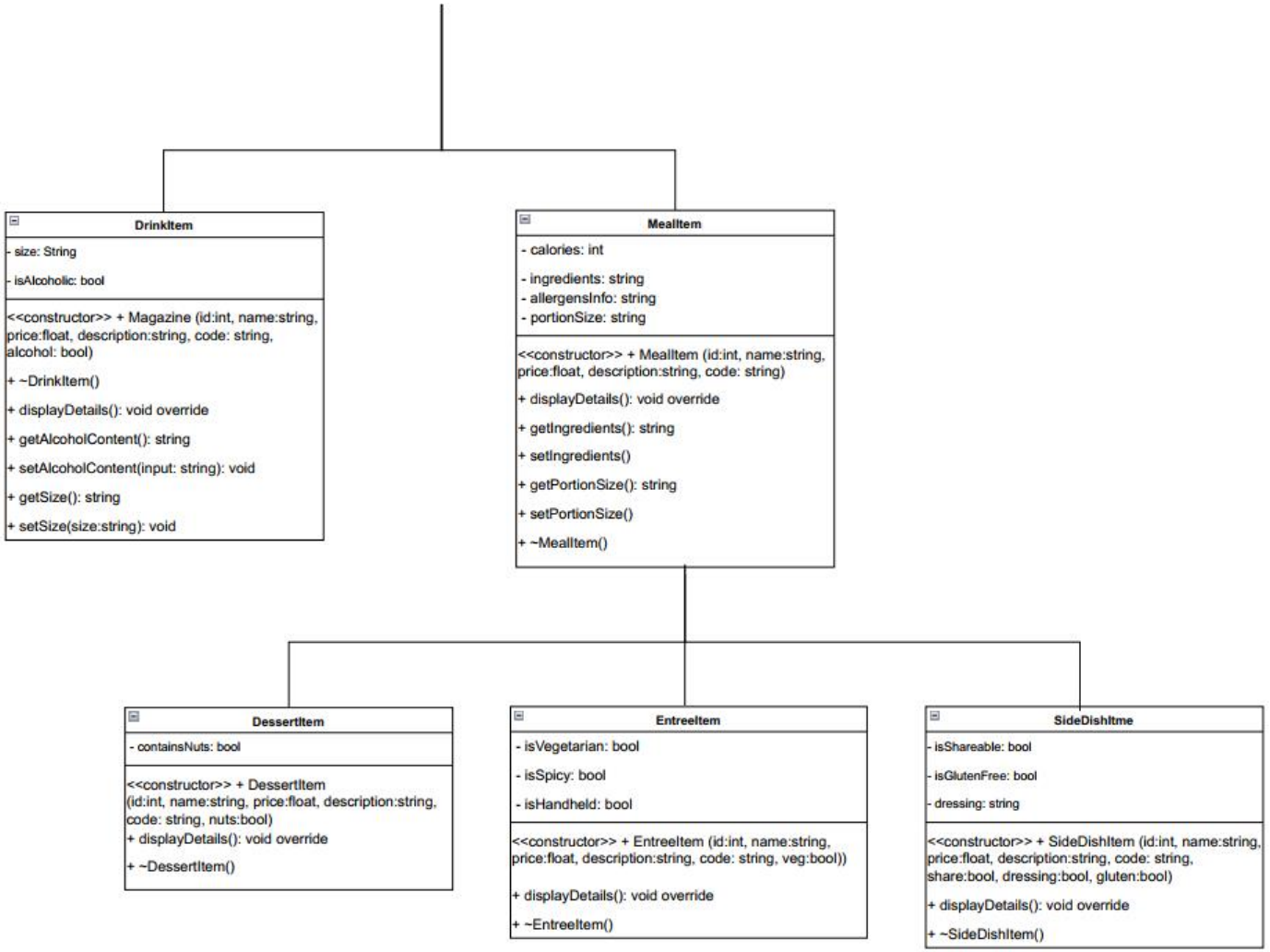
Polymorphism is used extensively in this class where pointers to RestuarantItem are used to simplify display of various types of meals and drinks

3f Customer Class (Derived Class)

The Customer class which has a protected access to its Base class is utilized to manage customer orders. It contains data members for capturing customer information such as name, address etc. For order management, a multimap is defined to hold customer orders. This is made static so that it has global scope allowing one shared instance of a customer order accessible globally.

4 UML Class Diagram





UML CLASS DIAGRAM

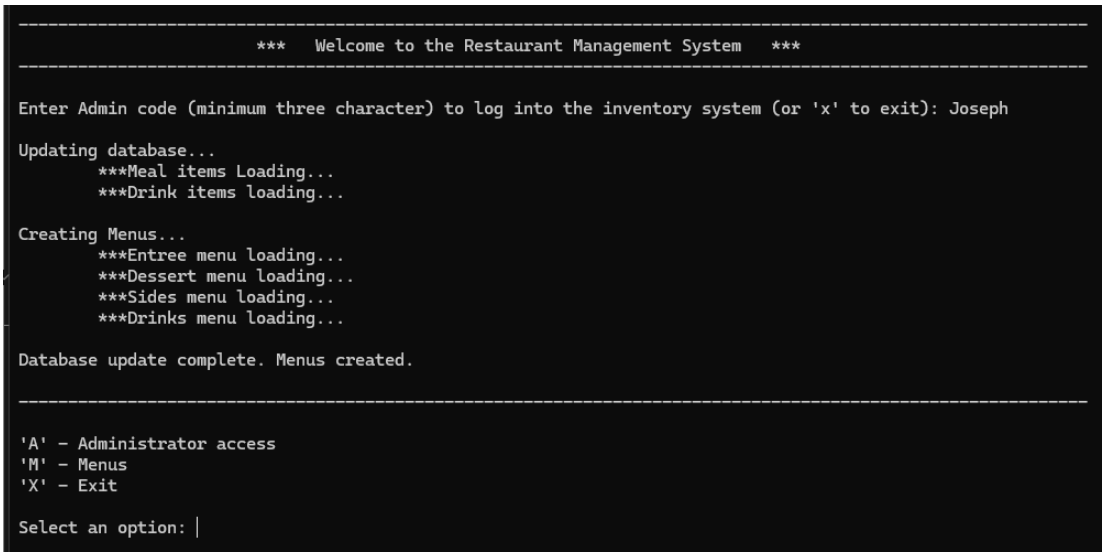
LIBRARY MANAGEMENT SYSTEM

5 User Interface

5a Login and loading of database

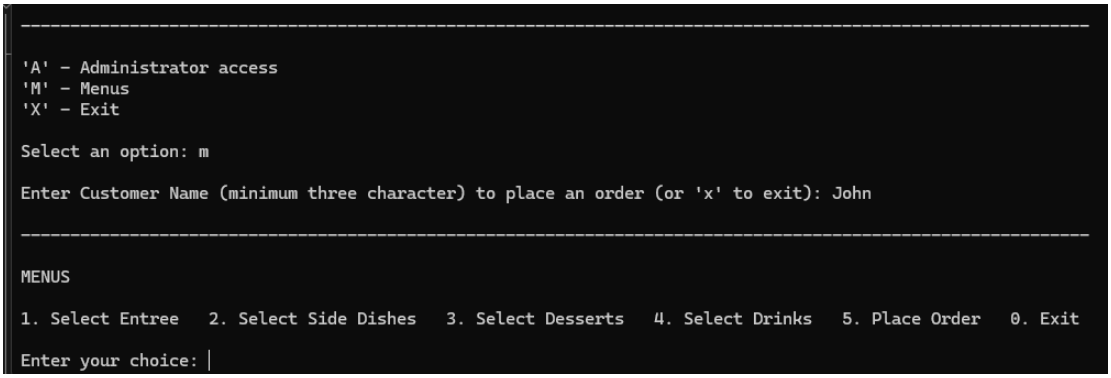
At startup, an admin login is requested to activate the system. This triggers methods to prepopulate meal and drink items as well as food and drinks menus.

Login interface



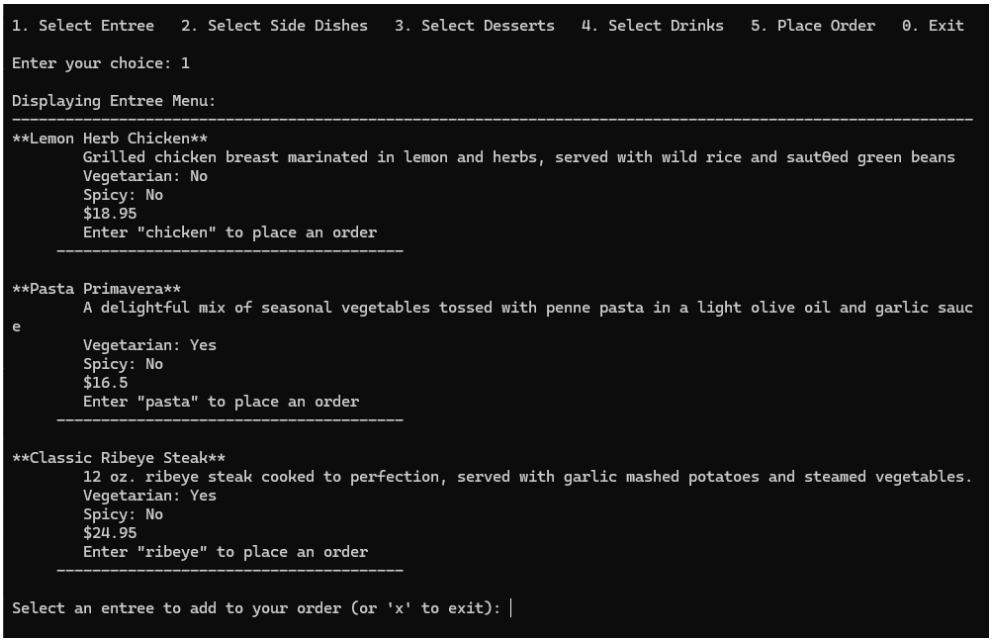
5b Customer Interaction

Customer Login and Menu Navbar

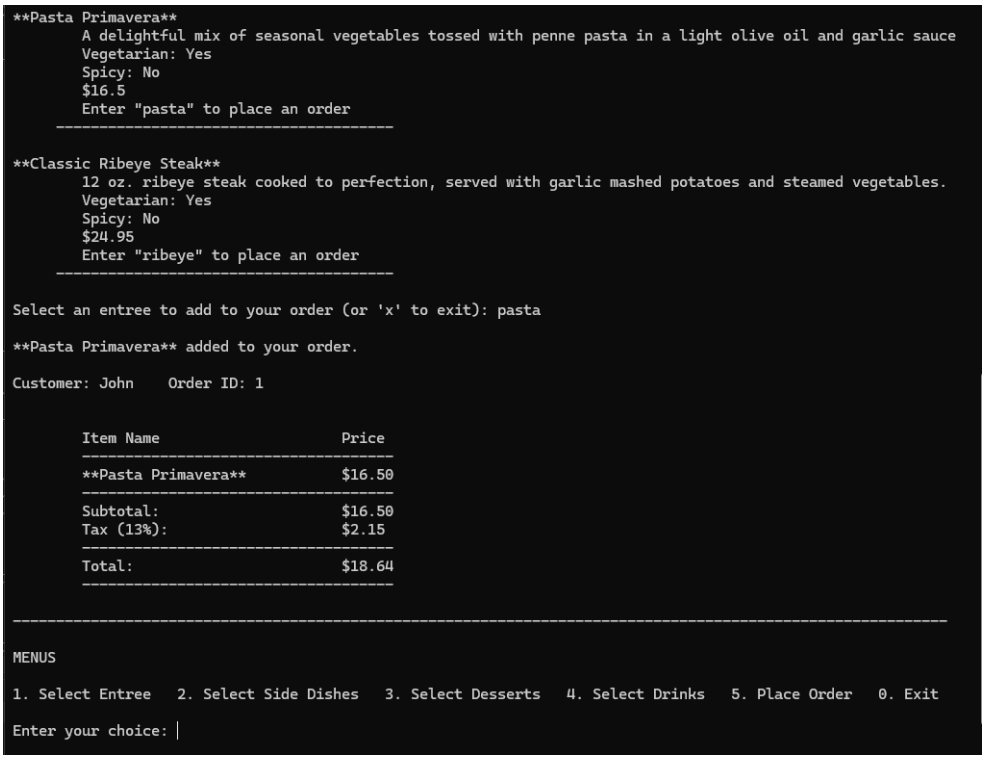


Selecting ‘M’ from the available options navigates to the Customer Interface and displays a message to get the user’s name. A navigation bar of menu options is also displayed.

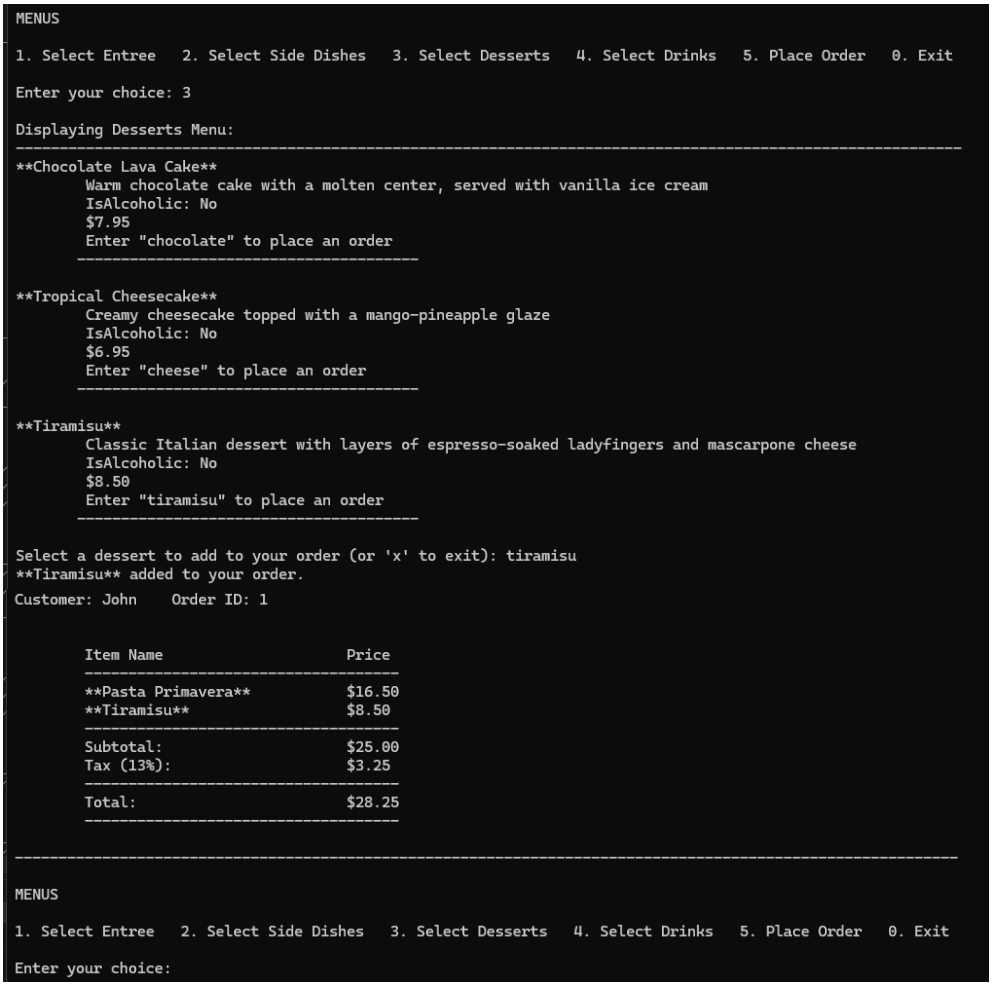
Meal and Drinks Selection, Order Detail



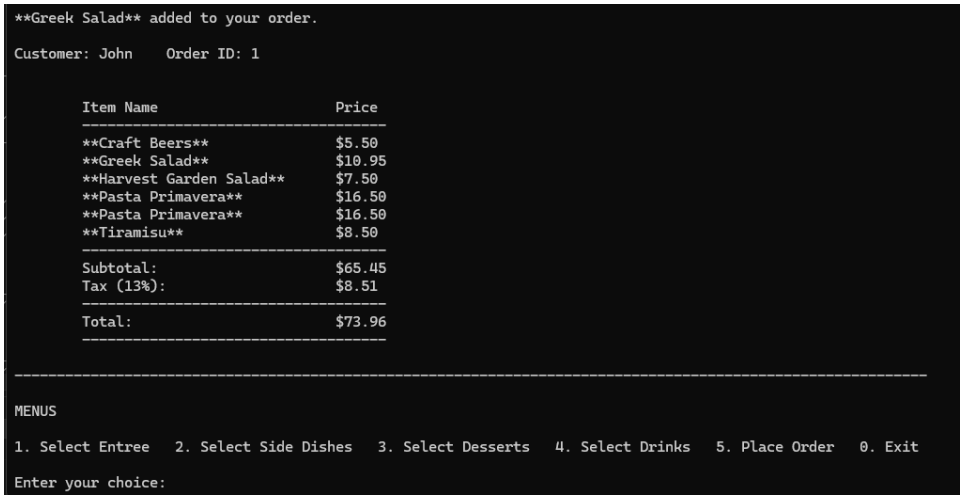
Once a menu option is selected, menu items are displayed. The user is asked to enter a meal.



After entering a meal type, order details are displayed showing selected meal, price, tax and total. The user can select additional meal items.



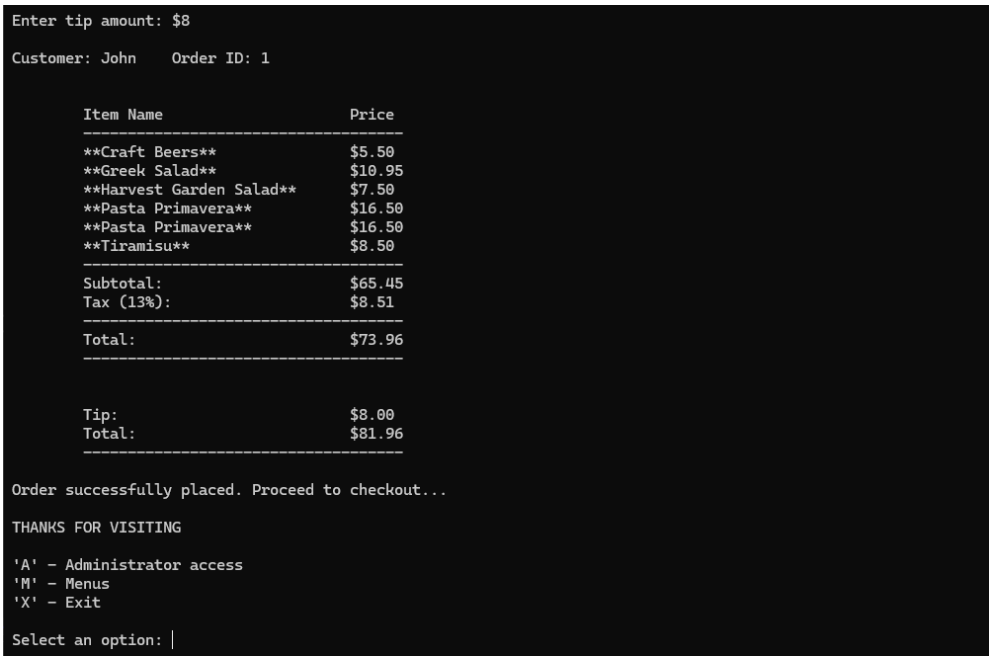
The order detail is update dynamically as the user adds items to the order



The order detail updated dynamically with additional items added to the order

Place Order / Check Out

Option 5 – place order, displays the invoice showing order details. There is a prompt to enter a tip amount



Once the tip amount is entered. The finalized invoice is display and a message to proceed to checkout. The loop goes back to the user login options

5c Admin manager

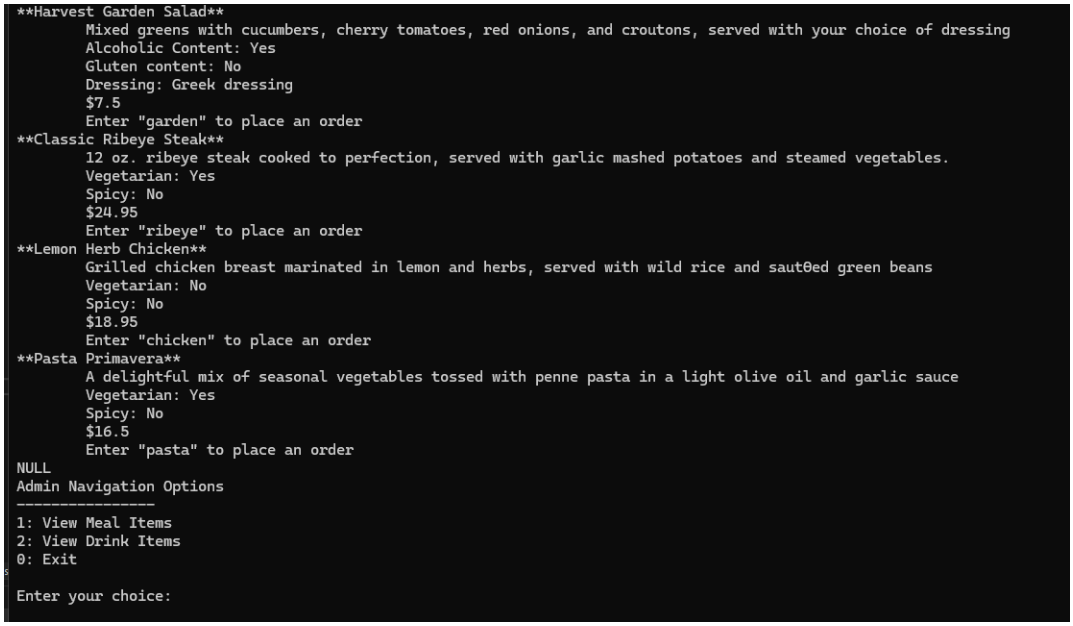
The admin manage features implemented so far demonstrate that the linked list is function as it displays the full meals list and drinks list. Further implementation can add features to manage the list items with the already defined functions in the Linkedlist class.

Login / Admin interface

```
------------------------------------------------------------------------------------------------------------
                        ***   Welcome to the Restaurant Management System   ***
------------------------------------------------------------------------------------------------------------

Enter Admin code (minimum three character) to log into the inventory system (or 'x' to exit): Adam

Updating database...
        ***Meal items Loading...
        ***Drink items loading...

Creating Menus...
        ***Entree menu loading...
        ***Dessert menu loading...
        ***Sides menu loading...
        ***Drinks menu loading...

Database update complete. Menus created.

------------------------------------------------------------------------------------------------------------

'A' - Administrator access
'M' - Menus
'X' - Exit

Select an option: A
Enter Admin code (minimum three character) to log into the inventory system (or 'x' to exit): Adam

Admin Navigation Options
----------------
1: View Meal Items
2: View Drink Items
0: Exit

Enter your choice:
```

Selecting ‘A’ from the available options navigates to the Admin Interface and displays a message to get the user’s name. Admin navigation options are displayed.

```
----------------
1: View Meal Items
2: View Drink Items
0: Exit

Enter your choice: 1
Existing Meals in the Retaurant:
**Tiramisu**
        Classic Italian dessert with layers of espresso-soaked ladyfingers and mascarpone cheese
        IsAlcoholic: No
        $8.5
        Enter "tiramisu" to place an order
**Tropical Cheesecake**
        Creamy cheesecake topped with a mango-pineapple glaze
        IsAlcoholic: No
        $6.95
        Enter "cheese" to place an order
**Chocolate Lava Cake**
        Warm chocolate cake with a molten center, served with vanilla ice cream
        IsAlcoholic: No
        $7.95
        Enter "chocolate" to place an order
**Greek Salad**
        Mixed greens with kalamata olives, feta cheese, red onions, and cucumbers, served with a Greek dressing
        Alcoholic Content: Yes
        Gluten content: Yes
        Dressing: Ranch dressing
        $10.95
        Enter "greek" to place an order
**Caesar Supreme Salad**
        Crisp romaine lettuce, parmesan cheese, and croutons tossed in a creamy Caesar dressing
        Alcoholic Content: No
        Gluten content: No
        Dressing: Thousand Island dressing
        $9
        Enter "ceaser" to place an order
```



Display of meals list from the admin interface

II Task 2 (Conceptual & Report)

This section reviews the performance of the reservation system used by the code in task 1 with a view to analyzing the system's efficiency, scalability, and reliability. It will review the data structures and algorithms employed, assess their impact on performance and provide recommendations for improvement including any potential trade-offs involved.

1 Review of Existing Data Structures and Algorithms

The review will focus on performance and scalability in terms of increased traffic and volume. Average time complexity will reviewed for more efficient systems for searching and sorting data. Space complexity will be review in terms of the amount of memory space required by an algorithm to execute, as a function of the size of the input data. Big O(n) will be used for the evaluation for insertion and searching operations.

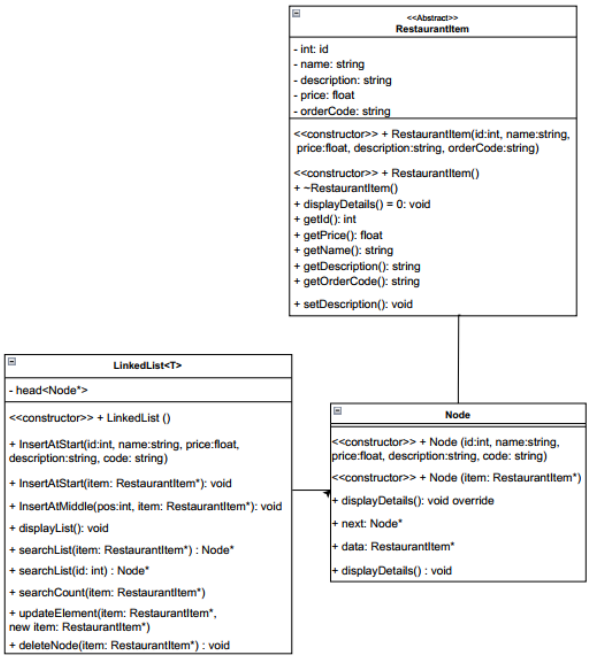
2 LinkedList for managing Meals and Drinks

2a Description of Current Scenario

A linked list data structure sits at the top hierarchy of data management for all restaurant items. This is defined in the “LinkedList” class with operations defined in the class including insertion, deletion, display and search functions. The linked list is employed mainly in the MenuManager class to manage overall listings of meal and drink items.

This allows for efficient dynamic insertion and deletion of elements at any position, which is useful for managing frequently updated inventory. It also easily accommodates varying numbers of items without the need for resizing or reallocation.

The data to be inserted in the linked list is defined in a “Node” which inherits from the RestaurantItem class. This enables constructor initialization of restaurant items in the linked lists if required. It has 2 data members which are both pointers, one points to Node type and the other points to RestaurantItem type.



2b Recommendations

The recommendation is to replace the linked list with a hash table since there is a requirement for quick lookup, insertion and deletion of meal items. The hash table will map keys to values for highly efficient lookup of O(1) average-case complexity.

Summary of Algorithm Analysis for Existing Linked list and Recommendations:

	Operations	Average Case O(n)
1.	Inserting an item at the start of the linked list:	
	Existing Design: - Involves creating a new node and updating the head pointer to point to this new node. It does not depend on the size of the list.	O(1)
	Hash Index recommendation: - Insertion involves calculating the hash index and placing the item in the corresponding slot.	O(1)
2.	Displaying all elements in the linked list:	
	Existing Design: - Requires traversing the entire linked list to display each element, therefore the time taken is proportional to the number of elements in the list.	O(n)
	Hash Index recommendation: - Similar to existing design	O(n)
3	Searching for a specific item in the linked list.	

	Existing Design: - The time taken is proportional to the number of elements in the list.	O(n)
	Hash Index recommendation: - With efficient hashing, on average, this is O(1). However, in the worst case it can degrade to O(n) if there is a long chain to traverse.	O(1)
4.	Updating an element in the linked list:	
	Existing Design: - This operation first involves searching for the node containing old_data, which is O(n). However, this is O91) once the data is found and updated.	O(n)
	Hash Index recommendation: - Updating an item involves searching for the item (O(1) on average) and replacing it.	O(1)

Overall, using a hash table provides better efficiency since it does not require transversal of the structure for displaying, searching and updating items leading to O(1) average complexity. The trade off in using a hash table is that there could be need for extra space that are potentially empty. This will be highly scalable and efficient data structure as the meal data grows.

3 Vectors for Managing Food Menus

3a Review of Current Scenario / Recommendations

Using vectors to manage food menus is advantageous for dynamic resizing, enabling flexibility for updating the menu adding or removing items without requiring manual memory management. The use of pointers to a base class of RestaurantItem* allows the vector to store different derived class objects, enabling polymorphic behavior.

Vectors allow for easy iteration and was used effectively in the searching and display functions for various items.

A custom class may beneficial where more complex operations can be developed for managing menus.

4 Multimap for Managing Customer Orders

4a Review of Current Scenario / Recommendations

A multimap (static multimap<string, RestaurantItem*>) is used for customer orders allowing efficient organization and retrieval of customer orders based on the name of the RestaurantItem which is the key in this instance. It allows for quick lookup and grouping of items by name, facilitating easy managing and displaying of orders. The

multimap allows for entries with the same key, which works well with scenarios where a customer might order multiple instances of the same item.

Since the matching pair is a pointer to the base class RestaurantItem, it enables polymorphic behaviour as it can be used for storing various derived classes in the same collection and avoids copying of objects since the objects are not passed by value.

This data is made static so that it has global scope which allows global visibility. This ensures that data is preserved after it is updated within any local scope in the main method. For example, after the customer finalizes their order, the calculation of payment is done in another scope using data previously entered by the customer.

Since orders require an associative container to manage chosen menu items, the multimap is a good choice. It is not anticipated that each customer will have an enormous amount of items in the same order so time and space complexity is not a major concern. Additional functionality can be added whereby customers are able to undo last item using a stack structure can beneficial.

III Task 3 (Conceptual & Research)

5 Evaluation of Security Measures

This section evaluates the security measures implemented in the payment processing system of the code above with a view to ensuring safety of customer transactions. The system's authentication mechanisms and potential vulnerabilities will be reviewed.

5a Review of Current Authentication

	Current Measures:
	- Input Validation: The code uses validation files for character, string, and integer inputs, which is good for ensuring that user inputs meet expected formats and constraints. - Regex for Password Policy: Regex is used for enforcing password policies, which is useful for ensuring strong passwords.
	Recommendations
	- Rate Limiting: The code can include rate limiting for login attempts to prevent brute force attacks. - Account Lockout: Introduction of account lockout mechanisms after a certain number of failed login attempts to mitigate automated attacks

5b Memory Management

	Current Measures:
	- There is inconsistent deallocation of memory created dynamically which can lead to memory leaks.
	Recommendations
	- Use of smart pointers - Consistent cleanup and proper deallocation in the class destructors

5c Evaluation of Exception Handling

	Current Measures:
	- The code does not implement try-catch blocks for several critical operations including object initialization and database loading at project start up.
	Recommendations
	- Implement try-catch blocks around critical sections to handle exceptions gracefully. - Specific Exception Types can be implemented to handle different error conditions as required. - Implement logging of errors when exceptions are caught to provide insights into issues during runtime.

5d Evaluation of Exit Points

	Current Measures:
	- There are several exit points in the code which can lead to inconsistent cleanup of resources and dangling pointers.
	Recommendations
	- Design for one exit point where dynamic resources can be cleaned up and resources deallocated

IV Conclusion

This project implements a Restaurant management system using Object Oriented Programming in C++ with a focus on inheritance, polymorphism, encapsulation and error handling. The project is the design of an interactive menu for a new restaurant where customers can view the menu, place orders, and customize their selections.

Design of classes utilize class hierarchy and inheritance. A base class RestaurantItem was created to encapsulate attributes common to all meal and drink items. Derived classes inherit from the base class and extend specific functionalities.

The RestaurantItem class used a pure virtual function, display(), to enforce that all derived classes implement their specific display logic. This approach demonstrated polymorphism allowing customized presentation of each item type while maintaining a common interface.

The MenuManager class manages the RestaurantItem using container data members to create collections and several member functions to manage addition and combination of data members as required.

Data structures used include linked lists for dynamic and efficient insertion and deletion of overall items. Vectors were utilized for managing menus with dynamic resizing, ensuring efficient access and manipulation of menu items and multimaps were used to manage orders where multiple items could be associated with a single customer.

V Presentation YouTube link

VI References

This section describes the documents and other sources from which information was gathered.

[1]	Walter Savitch, Absolute C++
[2]	Deitel and Deitel, C++ How to Program
[4]	M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.